



```
RRRRRRRR  MM      MM      000000  RRRRRRRR  EEEEEEEEE  CCCCCCCC  LL      CCCCCCCC  KK      KK
RRRRRRRR  MM      MM      000000  RRRRRRRR  EEEEEEEEE  CCCCCCCC  LL      CCCCCCCC  KK      KK
RR      RR  MMMM  MMMM  00      00  RR      RR  EE      CC      LL      CC      KK      KK
RR      RR  MMMM  MMMM  00      00  RR      RR  EE      CC      LL      CC      KK      KK
RR      RR  MM  MM  MM  00      0000  RR      RR  EE      CC      LL      CC      KK      KK
RR      RR  MM  MM  MM  00      0000  RRRRRRRR  EEEEEEEEE  CC      LL      CC      KK      KK
RRRRRRRR  MM      MM      00  00  00  RRRRRRRR  EEEEEEEEE  CC      LL      CC      KKKKKK
RRRRRRRR  MM      MM      00  00  00  RRRRRRRR  EEEEEEEEE  CC      LL      CC      KKKKKK
RR      RR  MM      MM      0000  00  RR      RR  EE      CC      LL      CC      KK      KK
RR      RR  MM      MM      0000  00  RR      RR  EE      CC      LL      CC      KK      KK
RR      RR  MM      MM      00      00  RR      RR  EE      CC      LL      CC      KK      KK
RR      RR  MM      MM      00      00  RR      RR  EE      CC      LL      CC      KK      KK
RR      RR  MM      MM      000000  RR      RR  EEEEEEEEE  CCCCCCCC  LLLLLLLLLL  CCCCCCCC  KK      KK
RR      RR  MM      MM      000000  RR      RR  EEEEEEEEE  CCCCCCCC  LLLLLLLLLL  CCCCCCCC  KK      KK
                                     ....
                                     ....
                                     ....
                                     ....

LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
```

(2)	183	DECLARATIONS
(3)	214	RMSLOCK AND RMSQUERY_LCK
(5)	456	DO ENQ
(6)	659	SCAN
(7)	766	RUSCAN
(8)	881	FLB_SCAN
(9)	901	PRSCAN
(10)	972	GET_RLB AND RESET_RLB
(11)	1075	RMSUNLOCK AND RMSUNLOCKALL
(12)	1263	RMS\$SAVE_FL
(13)	1299	RMSRU_UNLOCK



```
0000 1 $BEGIN RMORECLCK,000,RMSRMS0,<RECORD LOCK LIST (RLB) PROCESSING>
0000 2
0000 3
0000 4 *****
0000 5 *
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
0000 8 * ALL RIGHTS RESERVED. *
0000 9 *
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
0000 15 * TRANSFERRED. *
0000 16 *
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
0000 19 * CORPORATION. *
0000 20 *
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
0000 23 *
0000 24 *
0000 25 *****
0000 26
0000 27 ++
0000 28 Facility: rms32
0000 29
0000 30 Abstract:
0000 31 This module performs all the functions needed to implement
0000 32 and process the record lock list (rlb).
0000 33
0000 34 Environment:
0000 35 Star processor running starlet exec.
0000 36
0000 37 Author: E. H. Marison, creation date: 28-SEP-1977
0000 38
0000 39 Modified By:
0000 40
0000 41 V03-014 JEJ0043 J E Johnson 21-Jun-1984
0000 42 Tweak the instruction stream for a little
0000 43 performance boost.
0000 44
0000 45 V03-013 JWT0160 Jim Teague 29-Feb-1984
0000 46 Remove calls to RMS$DEALLEN.
0000 47
0000 48 V03-012 JWT0141 Jim Teague 11-Nov-1983
0000 49 Change IFBSV_RUM to IFBSV_ONLY_RU
0000 50
0000 51 V03-011 DAS0004 David Solomon 27-Jun-1983
0000 52 Correct typo in V03-010.
0000 53
0000 54 V03-010 KPL0008 Peter Lieberwirth 21-Jun-1983
0000 55 Set LCK$M_PROTECT if file can be recovery unit journaled.
0000 56 Set LCK$M_RECOVER if recovery unit is being recovered. These
0000 57 flags are used to coordinate failover between the lock manager
```

0000 58 : and the RCP.  
0000 59 :  
0000 60 : V03-009 KPL0007 Peter Lieberwirth 20-Jun-1983  
0000 61 : Change some references to JNLFLG to JNLFLG2.  
0000 62 :  
0000 63 : V03-008 JWH0198 Jeffrey W. Horn 21-Mar-1983  
0000 64 : Restructure RULOCK list so that the RLBs hang off  
0000 65 : of FLBs (File lock blocks).  
0000 66 : Add RMSSAVE FL to save the file lock on the RULOCK  
0000 67 : list in a FCB.  
0000 68 :  
0000 69 : V03-007 DAS0003 David Solomon 24-Feb-1983  
0000 70 : Add timeout on record lock wait capability.  
0000 71 :  
0000 72 : V03-006 JWH0181 Jeffrey W. Horn 03-Feb-1983  
0000 73 : Add check for IFBSV\_RU\_RLK to QUERY.  
0000 74 :  
0000 75 : V03-005 JWH0175 Jeffrey W. Horn 26-Jan-1983  
0000 76 : Fix bad branch destination in RUSCAN when the RLB  
0000 77 : is not found.  
0000 78 : Always make sure IFBSV\_NO\_Q\_WAIT is clear when exiting  
0000 79 : RMSQUERY\_PROC and RMSQUERY\_LCK.  
0000 80 :  
0000 81 : V03-004 JWH0169 Jeffrey W. Horn 10-Jan-1983  
0000 82 : If giving back a Recovery Unit held lock then return  
0000 83 : alternate status of OK\_RULK.  
0000 84 :  
0000 85 : Add RMSQUERY\_PROC to search for a lock on the RU list regardless  
0000 86 : of stream and then if not found join QUERY.  
0000 87 :  
0000 88 : For QUERY, if we had to do an \$ENQ, then always deque the  
0000 89 : lock, regardless of RU.  
0000 90 :  
0000 91 : If the bit IFBSV\_RU\_RLK is set then do not perform \$ENQs.  
0000 92 :  
0000 93 : If the bit IRBSV\_NO\_Q\_WAIT is set then do not wait on \$ENQs  
0000 94 : within QUERY.  
0000 95 :  
0000 96 : V03-003 JWH0001 Jeffrey W. Horn 19-Aug-1982  
0000 97 : Put in Recovery Unit Lock support:  
0000 98 :  
0000 99 : 1. If stream releases lock in RU hold lock  
0000 100 : in PIOSGL\_RULOCK list.  
0000 101 :  
0000 102 : 2. If same steam trys to re-access lock  
0000 103 : released in that RU, give it back after  
0000 104 : conversion if necessary.  
0000 105 :  
0000 106 : 3. Put in RMSRU\_UNLOCK routine to release  
0000 107 : all locks held for the durration of an RU.  
0000 108 :  
0000 109 : 4. RLBSL\_OWNER now contains the value of  
0000 110 : IRBSL\_IDENT for the owning stream, which  
0000 111 : is a Unique (for the life of the process)  
0000 112 : identifier for each stream.  
0000 113 :  
0000 114 : V03-002 KBT0307 Keith B. Thompson 25-Aug-1982



0000 115 : Reorganize psects  
0000 116 :  
0000 117 : V03-001 CDS0001 C Saether 1-Mar-1982  
0000 118 : Save R2 when stalling for a record lock.  
0000 119 :  
0000 120 : V02-011 KPL0006 Peter Lieberwirth 21-Oct-1981  
0000 121 : Add additional entry points so that query\_lck and unlock  
0000 122 : will return a RNL status in those places where a REA lock  
0000 123 : is held and the caller expects to get away with doing an  
0000 124 : update or delete after a get or find (that only applied a  
0000 125 : REA lock). This is important because several streams can  
0000 126 : hold a REA lock on a single record, so if any can update  
0000 127 : the record, consistency is lost.  
0000 128 :  
0000 129 : This wasn't a problem before because REA locks weren't  
0000 130 : really being applied properly (see next paragraph), and REA  
0000 131 : will now be permitted in files opened for write access.  
0000 132 :  
0000 133 : Fix implementation of REA lock; it should map to PR (protected  
0000 134 : read) not PW (protected write). If both REA and RLK are set,  
0000 135 : REA takes precedence.  
0000 136 :  
0000 137 : Remove bugcheck on DEQ\_S failure, it doesn't do anything good.  
0000 138 :  
0000 139 : Fix some commentary.  
0000 140 :  
0000 141 : V02-010 kpl0005 Peter Lieberwirth 30-Sep-1981  
0000 142 : Always release curbdb on record lock stall.  
0000 143 :  
0000 144 : V02-009 kpl0004 Peter Lieberwirth 3-Aug-1981  
0000 145 : Make the following changes:  
0000 146 :  
0000 147 : 1. move RLBS to a list off the IRAB from the IFAB  
0000 148 : 2. zero CURBDB so GET doesn't try to deaccess again  
0000 149 : on errors (WAT only)  
0000 150 : 3. when WAT not taken, deallocate the sync efn as well  
0000 151 : as set it  
0000 152 : 4. redesign and fix bugs regarding RRL and REA  
0000 153 : 5. remove RMSUNLOCK\_ALT entry point since its no longer  
0000 154 : needed  
0000 155 :  
0000 156 : V02-008 mcn0006 Maria del C. Nasr 23-Jul-1981  
0000 157 : record id size changes from a byte to a word  
0000 158 :  
0000 159 : V02-007 kpl0003 Peter Lieberwirth 7-Jul-1981  
0000 160 : Add testpoints to count number of times RMSLOCK and  
0000 161 : RMSQUERY\_LCK are called. Also add a testpoint to see  
0000 162 : how many times we do a wait on a record lock conflict.  
0000 163 : (This last depends on user setting the ROP WAT bit.)  
0000 164 :  
0000 165 : V02-006 kpl0002 Peter Lieberwirth 5-Jan-1981  
0000 166 : Rewrite to use \$eq/\$deq to lock and unlock records.  
0000 167 : rm\$query\_lock can now return ok rrl if ROP function  
0000 168 : RRL is specified and record is locked against readers.  
0000 169 :  
0000 170 : V02-005 REFORMAT Maria del C. Nasr 24-Jul-1980  
0000 171 :

RMORECLK  
V04-000

RECORD LOCK LIST (RLB) PROCESSING H 10

16-SEP-1984 00:32:06  
5-SEP-1984 16:22:15

VAX/VMS Macro V04-00  
[RMS.SRC]RMORECLK.MAR;1

Page 4  
(1)

0000	172	:	V004	RAN0003	R A NEWELL	9-nov-1978	10:14
0000	173	:		file sharing code enhancements			
0000	174	:					
0000	175	:		Revision History:			
0000	176	:					
0000	177	:		L F LAVERDURE,	9-oct-1978 17:16		
0000	178	:		add shared file code			
0000	179	:					
0000	180	:	--				
0000	181	:					

```

0000 183      .SBTTL DECLARATIONS
0000 184
0000 185 :
0000 186 : Include Files:
0000 187 :
0000 188
0000 189      $RABDEF
0000 190      $RLBDEF
0000 191      $IRBDEF
0000 192      $IFBDEF
0000 193      $RMSDEF
0000 194      $$FSBDEF
0000 195      $$SDEF
0000 196      $ENQDEF
0000 197      $LCKDEF
0000 198      $FLBDEF
0000 199
0000 200 :
0000 201 : Macros:
0000 202 :
0000 203 :
0000 204 : Equated Symbols:
0000 205 :
0000 206
0000 207      ROP      =      RAB$L_ROP*8
0000 208
0000 209
0000 210 :
0000 211 : Own Storage:
0000 212 :
00000020

```



```
0000 214 .SBTTL RMSLOCK AND RMSQUERY_LCK
0000 215
0000 216 :++
0000 217
0000 218 RMSLOCK - make entry in the lock list for specified record
0000 219 RMSQUERY_LCK - search rlb for specified record and report status
0000 220
0000 221 Calling sequence:
0000 222 bsbw rm$lock
0000 223 bsbw rm$query_lck
0000 224
0000 225
0000 226 Input Parameters:
0000 227
0000 228 r11 impure area address
0000 229 r10 ifab address
0000 230 r9 irab address *** please note always irab ***
0000 231 r8 rab address
0000 232 r1 1'st and 2'nd word of record's rfa
0000 233 r2 3'rd word of record's rfa
0000 234 seq f.o. offset (always positive value)
0000 235 relative f.o. always 0
0000 236 index f.o. low byte = record id
0000 237
0000 238
0000 239 Implied Input:
0000 240 rm$lock
0000 241 the wat bit in rop (ie queue the request if it can't
0000 242 be granted immediately)
0000 243 the tmo bit in rop (if WAT is set, wait for a specific amount
0000 244 of time before returning timeout error).
0000 245 the rlk bit in rop (ie lock for write, allow readers)
0000 246 the rea bit in rop (ie lock for read, allow readers)
0000 247 rm$query_lck
0000 248 the rrl bit in rop (ie read record regardless of lock)
0000 249
0000 250 Output Parameters:
0000 251
0000 252 r3 is destroyed
0000 253
0000 254 r0 internal rms status code
0000 255 rm$lock:
0000 256 rms$_suc&^xffff record lock entry made
0000 257 rms$_ok_wat&^xffff record lock entry was made, but we had
0000 258 to wait to get it, caller must reaccess
0000 259 buffer
0000 260 rms$_ok_alk&^xffff record was already locked by caller
0000 261 rms$_rlk&^xffff record is locked by another
0000 262 process-stream
0000 263 rms$_dme&^xffff could not get space for new rlb block
0000 264 rms$_tmo&^xffff record lock timed out
0000 265
0000 266 rm$query_lck:
0000 267 rms$_suc&^xffff record not locked
0000 268 rms$_ok_alk&^xffff record was already locked by caller
0000 269 rms$_ok_rlk&^xffff record is locked by another
0000 270 process-stream but read is allowed
```

```
0000 271 : rms$_ok_rrl&^xffff record is locked by another
0000 272 : process-stream, but RRL overrides lock
0000 273 : rms$_rlk&^xffff record is locked by another
0000 274 : process-stream
0000 275 :
0000 276 :
0000 277 : Side Effects:
0000 278 :
0000 279 : If success code rms$_ok_wat is returned from RMSLOCK, record was
0000 280 : successfully locked, but access to the buffer was given up to do
0000 281 : a stall. The caller must reaccess the buffer.
0000 282 :
0000 283 : --
0000 284 :
0000 285 : RMSLOCK::
0000 286 : $TSTPT LOCK ; count this call
0006 287 :
0A 00A2 CA 02 E1 0006 288 BBC #IFBSV_RUP,IFBSB_JNLFLG2(R10),5$ ; branch if not in RU
028E 30 000C 289 BSBW RUSCAN ; scan RU lock list
5D 50 E8 000F 290 BLBS R0,50$ ; get out if one found
50 50 D5 0012 291 TSTL R0 ; was it an error instead?
59 12 0014 292 BNEQ 50$ ; yes, get out.
0239 30 0016 293 5$: BSBW SCAN ; scan lock list
0C 50 E9 0019 294 BLBC R0,10$ ; only success codes are ok_alk and ok_rlk
50 8039 8F B1 001C 295 CMPW #<RMS$_OK_ALK&^XFFFF>,R0 ; already locked?
4C 13 0021 296 BEQL 50$ ; branch if yes
0023 297 RMSERR RLK ; otherwise change it to rlk
53 D4 0028 298 10$: CLRL R3 ; if we get here don't want this rlb
50 81A0 8F B1 002A 299 CMPW #<RMS$_RNL&^XFFFF>,R0 ; record not in local lock list?
3E 12 002F 300 BNEQ 50$ ; exit if record is locked
034F 30 0031 301 BSBW GET_RLB ; find an RLB
38 50 E9 0034 302 BLBC R0,50$ ; branch on any error
0037 303
05 68 2C E1 0037 304 BBC #RABSV_LV2+ROP,(R8),15$ ; propagate LV2 bit to RLB
003B 305 SSB #RLBSV_LV2,RLBSB_FLAGS(R3)
0040 306
05 68 22 E1 0040 307 15$: BBC #RABSV_REA+ROP,(R8),20$ ; is it a REA type lock? branch if no
03 E3 0044 308 BBBS #RLBSV_PR,- ; map REA to protected read
09 0B A3 0046 309 RLBSB_FLAGS(R3),30$
0049 310
05 68 33 E1 0049 311 20$: BBC #RABSV_RLK+ROP,(R8),30$ ; is it a RLK type lock? branch if no
02 E3 004D 312 BBBS #RLBSV_PW,- ; map RLK to protected write
00 0B A3 004F 313 RLBSB_FLAGS(R3),30$
0052 314
0052 315 :+
0052 316 : Save record lock wait/timeout information in RLB. If the ROP WAT
0052 317 : bit is not set, don't even look at the TMO bit.
0052 318 :-
0052 319
13 68 31 E1 0052 320 30$: BBC #RABSV_WAT+ROP,(R8),40$ ; Should we wait on record lock?
00 00 E3 0056 321 BBBS #RLBSV_WAIT,- ; Yes, propagate bit to RLB.
00 0B A3 0058 322 RLBSB_FLAGS(R3),35$
005B 323
0A 68 39 E1 005B 324 35$: BBC #RABSV_TMO+ROP,(R8),40$ ; Is a timeout specified?
07 E3 005F 325 BBBS #RLBSV_TMO,- ; Propagate bit to RLB.
00 0B A3 0061 326 RLBSB_FLAGS(R3),37$
1F A8 90 0064 327 37$: MOVB RABSB_TMO(R8),- ; Store timeout value in RLB.
```

RMORECLCK  
V04-000

RECORD LOCK LIST (RLB) PROCESSING L 10  
RMSLOCK AND RMSQUERY\_LCK

16-SEP-1984 00:32:06 VAX/VMS Macro V04-00  
5-SEP-1984 16:22:15 [RMS.SRC]RMORECLCK.MAR;1

Page 8  
(3)

0A A3		0067	328		RLBSB_TMO(R3)	
		0069	329			
00BA	30	0069	330 40\$:	BSBW	DO ENQ	: lock the record
00A6	31	006C	331	BRW	RR	: go check for read-regardless
	05	006F	332 50\$:	RSB		: return all status to caller



```
0070 334 :++
0070 335 : RMSQUERY_LCK
0070 336 :
0070 337 : If the record is not locked locally, see if its locked by
0070 338 : another process by requesting a lock on it. If the lock
0070 339 : is granted, the record may be read. Also, immediately unlock
0070 340 : the record if lock granted, so extraneous junk doesn't fill
0070 341 : up the lock database.
0070 342 :
0070 343 : RMSQUERY_HARD
0070 344 :
0070 345 : Same as QUERY_LCK, but map OK_ALK when lock is REA type to RNL so
0070 346 : any writers of the file holding a REA lock on the record can't get
0070 347 : away with updating or deleting it.
0070 348 :
0070 349 : Algorithmn for query_lock
0070 350 :
0070 351 : first try PR - if this succeeds, it means there
0070 352 : was no lock, and its OK to read.
0070 353 :
0070 354 : if PR fails, it means either an EX or PW lock is
0070 355 : held on the record, so try CR, with WAIT if the
0070 356 : user said to. If CR succeeds, then the lock must
0070 357 : have been PW, so its OK to read.
0070 358 :
0070 359 : Also, read-regardless of lock (RRL) is handled here. If all
0070 360 : indications are that the record is locked, then if RRL is
0070 361 : specified, access to that record is permitted.
0070 362 :
0070 363 :
0070 364 : RMSQUERY_PROC
0070 365 :
0070 366 : Scan RU Lock List for lock regardless of stream, if found
0070 367 : return OK_RULK otherwise join RMSQUERY_LCK code.
0070 368 :
0070 369 :--
0070 370 :
0070 371 : RMSQUERY_HARD::
0070 372 : $STPT QUERY_LCK ; count this call
0070 373 : BSBW SCAN ; scan for record
0070 374 : CMPW #<RMS$_OK_ALK&^XFFFF>,R0 ; is record locked by caller?
0070 375 : BNEQ 10$ ; if NEQ no
0070 376 : BBC #RLB$_PR,- ; yes, but is it only REA?
0070 377 : RLBSB_FLAGS(R3),10$ ; branch if not REA
0070 378 : RMSERR RNL ; map OK_ALK to RNL if its locked REA
0070 379 : 10$: RSB ; return to caller
0070 380 :
0070 381 : RMSQUERY_PROC::
0070 382 : $STPT QUERY_LCK ; count this call
0070 383 : BSBW PRSCAN ; scan RU list for lock
0070 384 : BLBC R0,RMSQUERY_LCK ; continue with Query lock if not there
0070 385 : RMSSUC OK_RULK ; set alternate success
0070 386 : CSB #IRBSV_NO_Q_WAIT,(R9) ; make sure this bit is clear
0070 387 : RSB
0070 388 :
0070 389 : RMSQUERY_LCK::
0070 390 : $STPT QUERY_LCK ; count this call
```

50 01D9 30 0076 373  
8039 8F B1 0079 374  
0A 12 007E 375  
03 E1 0080 376  
05 0B A3 0082 377  
0085 378  
008A 379  
008B 380  
008B 381  
008B 382  
02B8 30 0091 383  
0A 50 E9 0094 384  
0097 385  
009C 386  
05 00A0 387  
00A1 388  
00A1 389  
00A1 390

```
50 01A8 30 00A7 391 BSBW SCAN ; scan for record
81A0 8F B1 00AA 392 CMPW #<RMS$_RNL&^XFFFF>,R0 ; if RNL, record may be locked by
; another process
20 12 00AF 393 ; return status of scan
03 E0 00B1 394 BNEQ 10$ ; get out if 'fake' record locking
1A 00A2 CA 00B3 395 BBS #IFBSV_RU_RLK,- ;
02C9 30 00B7 396 IFBSB_JNLFLG2(R10),10$ ;
14 50 E9 00BA 397 BSBW GET_RLB ; find an RLB to use
08 90 00BD 398 BLBC R0,T0$ ; pass along possible DME error
0B A3 00BF 400 MOVB #RLBSM_PR,- ;
51 DD 00C1 401 RLBSB_FLAGS(R3) ; ask only to read
61 10 00C3 402 PUSHL R1 ; save RFA across enq
51 8ED0 00C5 403 BSBB DO_ENQ ; go try to lock the record
0B 50 E9 00C8 404 POPL R1 ; restore RFA
0326 30 00CB 405 BLBC R0,20$ ; if error, go try CR
; got the lock, so give it up now
; permission to read record
; make sure this bit is clear
; return to caller
05 00D5 408 10$: CSB #IRBSV_NO_Q_WAIT,(R9)
00D6 409 RSB
02E3 30 00D6 410 20$: BSBW SETOWNRFA ; reset ownership and rfa
; try again for lock
02 90 00D9 411 MOVB #RLBSM_CR,- ;
0B A3 00DB 412 RLBSB_FLAGS(R3) ; try for concurrent read
00DD 413
00DD 414
00DD 415 ;+
00DD 416 ; Save record lock wait/timeout information in RLB. If the ROP WAT
00DD 417 ; bit is not set, don't even look at the TMO bit.
00DD 418 ; -
00DD 419
17 69 38 E4 00DD 420 30$: BBSC #IRBSV_NO_Q_WAIT,(R9),40$ ; branch if no queuing
13 68 31 E1 00E1 421 BBC #RABSV_WAT+ROP,(R8),40$ ; is queuing disabled
00E5 422 SSB #RLBSV_WAIT,-
00E5 423 RLBSB_FLAGS(R3)
0A 68 39 E1 00EA 424 BBC #RABSV_TMO+ROP,(R8),40$ ; Is a timeout specified?
00EE 425 SSB #RLBSV_TMO,- ; Propagate bit to RLB.
00EE 426 RLBSB_FLAGS(R3)
1F A8 90 00F3 427 MOVB RABSV_TMO(R8),- ; Store timeout value in RLB.
0A A3 00F6 428 RLBSB_TMO(R3)
00F8 429
2C 10 00F8 430 40$: BSBB DO_ENQ ; go try to lock the record
15 50 F9 00FA 431 BLBC R0,50$ ; branch on record lock error
50 DD 00FD 432 PUSHL R0 ; save status of lock operation
02F2 30 00FF 433 BSBW DEQUE_QUERY ; go unlock record we just locked
50 8ED0 0102 434 POPL R0 ; restore lock status
50 8061 8F B1 0105 435 CMPW #<RMS$_OK_WAT&^XFFFF>,R0 ; did we wait for the record?
05 13 010A 436 BEQL 45$ ; if eql we waited for the lock
010C 437 RMSSUC OK_RLK ; record locked, but ok to read
05 0111 438 RSB ; return to caller
02BE 30 0112 441 50$: BSBW RESET_RLB ; free the RLB
0115 442
0115 443 ;
0115 444 ; read regardless of lock:
0115 445 ;
0115 446 ; If the record is locked, then if the user specified RRL
0115 447 ; we'll return the record with the code OK_RRL.
```

50	82AA	8F	B1	0115	448	:			
		09	12	0115	449	RRL:			
05	68	23	E1	011A	450		CMPW	#<RMS\$_RLK&*XFFFF>,R0	: is the error RLK?
				011C	451		BNEQ	10\$	: if neq no, don't try read regardless
				0120	452		BBC	#RAB\$V_RRL+ROP,(R8),10\$	: has rrl been specified
				0125	453		RMSSUC	OK_RRL	: yes
			05		454	10\$:	RSB		: return to caller



```
0126 456 .SBTTL DO_ENQ
0126 457 :++
0126 458 DO_ENQ
0126 459 - build the enq on the stack, perform it, and handle any errors
0126 460 :
0126 461 Calling Sequence:
0126 462 bsbb do_enq
0126 463 :
0126 464 Input Parameters
0126 465 :
0126 466 r10 ifab address
0126 467 r9 irab address
0126 468 r8 rab/fab address (not needed by this routine, but
0126 469 must preserve this register)
0126 470 r3 rlb address
0126 471 :
0126 472 Output Parameters:
0126 473 :
0126 474 r0 status of enq
0126 475 rlb fields filled in: lock_id, lksb
0126 476 :
0126 477 Side Effects:
0126 478 :
0126 479 If success, record is locked. If status is OK_WAT, bucket has been
0126 480 deaccessed, and caller must reaccess.
0126 481 :
0126 482 :--
0126 483 :
0126 484 :
0126 485 DO_ENQ: ; perform the sys$enq
0126 486 RMSSUC ; assume success for the next check
0126 487 BBC #IFB$V_RU_RLK,IFB$B_JNLFLG2(R10),-
0126 488 5$ ; only do enq if sharing file
0126 489 SSB #RLB$V_FAKE,RLB$B_FLAGS(R3) ; set as 'fake' RLB
0126 490 RSB ; return to caller
0126 491 :
0126 492 5$: BBC #IFB$V_NORECLK,- ; only do enq if record locking
0126 493 (R10),TOS ;
0126 494 RSB ; return to caller
0126 495 :
0126 496 :
0126 497 ; Build RESNAM and descriptor for it on stack. Warning, if any parameters to
0126 498 SYS$ENQ are added or removed prior to RESNAM descriptor address, offset to
0126 499 RESNAM descriptor will change...
0126 500 :
0126 501 :
0126 502 10$: SUBL2 #16,SP ; make room for record RESAM and
0126 503 ; descriptor on stack
0126 504 MOVL #8,(SP) ; length of RESNAM in descriptor
0126 505 : RFA is only 6, but $enq optimizes 8
0126 506 MOVAL 8(SP),4(SP) ; address of RESNAM in descriptor
0126 507 MOVZWL RLB$W_RFA4(R3),8(SP) ; second longword of RESNAM is 3rd
0126 508 : word of RFA
0126 509 MOVL RLB$L_RFA0(R3),12(SP) ; first longword of RESNAM is 1st
0126 510 : longword of RFA
0126 511 :++
0126 512 :
```

00A2 CA 03 E1 0129 487  
06 012E 488  
05 012F 489  
01 33 E1 0134 490  
6A 0135 491  
05 0135 492 5\$:  
0137 493  
0139 494 RSB  
013A 495  
013A 496  
013A 497  
013A 498  
013A 499  
013A 500  
5E 10 C2 013A 501  
6E 08 D0 013D 502 10\$:  
013D 503  
0140 504  
04 AE 08 AE DE 0140 505  
08 AE 06 A3 3C 0145 506  
0C AE 0C A3 D0 014A 507  
014A 508  
014F 509  
014F 510  
014F 511 :++  
014F 512 :

```
014F 513 : Perform the sys$enq function, building the parameter list on the stack.
014F 514 :
014F 515 : First, verify assumptions about order of arguments on stack
014F 516 :
014F 517 :--
014F 518 :
014F 519 : ASSUME ENQ$_EFN EQ <ENQ$_LKMODE - 4>
014F 520 : ASSUME ENQ$_LKMODE EQ <ENQ$_LKSB - 4>
014F 521 : ASSUME ENQ$_LKSB EQ <ENQ$_FLAGS - 4>
014F 522 : ASSUME ENQ$_FLAGS EQ <ENQ$_RESNAM - 4>
014F 523 : ASSUME ENQ$_RESNAM EQ <ENQ$_PARID - 4>
014F 524 : ASSUME ENQ$_PARID EQ <ENQ$_ASTADR - 4>
014F 525 : ASSUME ENQ$_ASTADR EQ <ENQ$_ASTPRM - 4>
014F 526 : ASSUME ENQ$_ASTPRM EQ <ENQ$_BLKAST - 4>
014F 527 : ASSUME ENQ$_BLKAST EQ <ENQ$_ACMODE - 4>
014F 528 : ASSUME ENQ$_ACMODE EQ <ENQ$_PROT - 4>
014F 529 :
014F 530 : ASSUME ENQ$_NARGS EQ 11
014F 531 :
014F 532 : CLRQ -(SP) : let the protection and mode default
014F 533 : CLRL -(SP) : no blocking ast for records
014F 534 : PUSHL R9 : astprm = irab
014F 535 : PUSHAL W*RM$STALLAST : ast address
014F 536 : MOVL IFB$$_SFSB_PTR(R10),R0 : get SFSB address
014F 537 : BEQL 27$ : error if there is none
014F 538 : PUSHL SFSB$$_LOCK_ID(R0) : parent_id is SFSB lock id
014F 539 : PUSHAL 24(SP) : resnam_descriptor address
014F 540 : PUSHL #LCK$$_SYNCSTS!LCK$$_NOQUEUE!LCK$$_SYSTEM
014F 541 : : don't take ast if enq completes fast
014F 542 : : don't wait unless user tells us to
014F 543 : : lock is not to be qualified by group
014F 544 : BITB #<IFB$$_RU!IFB$$_ONLY_RU>,- : recovery unit journaled?
014F 545 : IFB$$_JNLFLG(R10)
014F 546 : BEQL 17$ : if EQL not marked for RU journaling
014F 547 : SSB #LCK$$_PROTECT,(SP) : lock is protected for failover
014F 548 :
014F 549 : ASSUME IFB$$_RU_RECVR EQ 0
014F 550 :
014F 551 : 17$: BLBC IFB$$_RECVRFGLS(R10),18$ : RU recovery in progress on this file?
014F 552 : SSB #LCK$$_RECOVER,(SP) : lock is interesting during failover
014F 553 :
014F 554 : ASSUME RLBS$$_WAIT EQ 0
014F 555 :
014F 556 : 18$: BLBC RLBS$$_FLAGS(R3),20$ : branch if not ok to wait
014F 557 : BICL2 #LCK$$_NOQUEUE,(SP) : wait for lock if not immediately
014F 558 : : available
014F 559 : 20$: BBC #RLBS$$_CONV,-
014F 560 : RLBS$$_FLAGS(R3),25$ : branch if not converting a lock
014F 561 : BICL2 #LCK$$_CONVERT,(SP) : set lock convert
014F 562 : 25$: PUSHAL RLBS$$_KSB(R3) : address of lock status block
014F 563 : PUSHL #LCK$$_EXMODE : assume exclusive lock for now
014F 564 : BBC #RLBS$$_PW,-
014F 565 : RLBS$$_FLAGS(R3),30$ : is it protected write?
014F 566 : MOVL #LCK$$_PWRITE,(SP) : make lkmode protected write
014F 567 : BRB 50$ : go allocate efn
014F 568 : 27$: BRW ERRENQ : branch aid
014F 569 : 30$: BBC #RLBS$$_PR,-
```

```
05 0B A3 019E 570 RLB$B_FLAGS(R3),40$ : is it protected read?
6E 03 DO 01A1 571 MOVL #LCK$R_PMODE,(SP) : make lkmode protected read
08 11 01A4 572 BRB 50$ : go allocate efn
01 E1 01A6 573 40$: BBC #RLB$V_CR,- :
03 0B A3 01A8 574 RLB$B_FLAGS(R3),50$ : is it concurrent read?
6E 01 DO 01AB 575 MOVL #LCK$R_CMODE,(SP) : make lkmode concurrent read
FE4F' 30 01AE 576 50$: BSBW RMS$SETEFN : allocate a synchronous event flag
01B1 577
00000000'9F 0B FB 01B1 578 CALLS #11,#SYS$ENQ : do the enq
5E 10 CO 01B8 579 ADDL2 #16,SP : pop RESNAM and its descriptor
75 50 E9 01BB 580 BLBC R0,110$ : branch on error
0689 8F 50 B1 01BE 581 CMPW R0,#SS$_SYNCH : synchronous completion?
11 12 01C3 582 BNEQ 90$ : no, go stall
01C5 583
01C5 584 $SETEF_S IRB$B_EFN(R9) : set event flag we didn't stall for
01CF 585 RMSSUC : indicate successful lock
05 01D2 586 RSB : and return
01D3 587
030B 31 01D3 588 80$: BRW ERRENQ : branch aid
01D6 589
01D6 590 :+
01D6 591 : If timeout on record lock specified, set up timer.
01D6 592 :--
01D6 593
07 E1 01D6 594 90$: BBC #RLB$V_TMO,- : If timeout not specified, skip this.
20 0B A3 01D8 595 RLB$B_FLAGS(R3),95$ :
FE22' 30 01DB 596 BSBW RMS$SET_LOCK_TMO : Set timer for lock.
1A 50 E8 01DE 597 BLBS R0,95$ : If successful, continue.
50 DD 01E1 598 PUSHL R0 : Save $SETIMR error status
50 8ED0 01E3 599 $DEQ_S LKID=RLB$L_LOCK_ID(R3) : Else $SETIMR failed; cancel $ENQ.
01F1 600 POPL R0 : Restore error status
01F4 601 RMSERR TMR,R1 : Unexpected $SETIMR error.
49 11 01F9 602 BRB 125$ : Go map error and exit.
01FB 603
1C BB 01FB 604 95$: PUSHR #*M<R2,R3,R4> : save registers
01FD 605
01FD 606 :++
01FD 607 : Release curbdb because we don't want the bucket locked while we are waiting
01FD 608 : (possibly for a long time) for the record. No one can even get in to unlock
01FD 609 : the record if we have the bucket locked.
01FD 610
01FD 611 : The extremely important assumption is made here that no STALL will be done in
01FD 612 : RMS$RELEASE. No bucket will be written for example. This call should only
01FD 613 : deaccess the buffer. If this assumption is invalid, then all $ENQ
01FD 614 : synchronization is blown because there aren't enough EFNs to go around.
01FD 615 :--
01FD 616
54 20 A9 DO 01FD 617 MOVL IRB$L_CURBDB(R9),R4 : point to current bdb
48 13 0201 618 BEQL NOBDB : error if there is none
0203 619
20 A9 D4 0203 620 CLRL IRB$L_CURBDB(R9) : zero CURBDB so error pats don't try
0206 621 : to release it again
0206 622 $TSTPT REC_WAT : count number of times WAT waits
020C 623
53 D4 020C 624 CLRL R3 : no flags for rm$release
FDEF' 30 020E 625 BSBW RMS$RELEASE : deaccess the buffer - no IO
FDEC' 30 0211 626 100$: BSBW RMS$STALL : await completion of enqueue
```



```
1C  BA 0214 627      POPR  #^M<R2,R3,R4>      ; restore registers
      0216 628
      0216 629
      0216 630      ;+
      0216 631      ; If a timer was still outstanding, cancel the request.
      0216 632      ; Note: if the timer fires after the BBCC instruction, but before the $CANTIM,
      0216 633      ; the timer AST routine will simply exit since the RLBSV_TIMER_INPROG flag will
      0216 634      ; be clear. If the timer fires before the BBCC instruction, the $DEQ will fail
      0216 635      ; with $$$_IVLOCKID, which is expected.
      0216 636      ;--
      0216 637
      00  E5 0216 638      BBCC  #RLBSV_TIMER_INPROG -      ; Continue if no $SETIMR outstanding.
OB 04 A3 0218 639      RLBSV_FLAGS2TR3),105$
      0218 640      $CANTIM_S -
      0218 641      -REQIDT=R3      ; Cancel timer request.
      0226 642
50  14 A3 3C 0226 643 105$: MOVZWL RLBSV_STATUS(R3),R0      ; copy enq status
      06 50 E9 022A 644      BLBC  R0,110$      ; branch on any error
      022D 645      RMSSUC OK_WAT      ; success, but we waited
      05 0232 646      RSB
      0233 647
      0233 648 110$: CMPL  R0, #$$$_ABORT      ; Was the lock request cancelled?
2C  50  D1 0233 649      BNEQ  120$      ; No; go map error.
      07 12 0236 650      RMSERR TMO      ; Primary status is timeout.
      08 11 0238 651      BRB  130$      ; Join exit code.
      023F 652 120$: RMSERR ENQ,R1      ; default to ENQ for RMSMAPERR
      FDB9' 30 0244 653 125$: BSBW  RMSMAPERR      ; note subroutine call, not branch!
      0189 30 0247 654 130$: BSBW  RESET_RLB      ; clear rlb, since we didn't get a lock
      05 024A 655      RSB      ; go return
      024B 656
      024B 657 NOBDB: RMSPBUG FTLS_NOCURBDB      ; there should be a current BDB
```

```
0252 659 .SBTTL SCAN
0252 660
0252 661 : **
0252 662 : SCAN
0252 663 - scan the rlb for the requested record (rfa <> 0)
0252 664 - scan for first record locked by caller if rfa = 0
0252 665 - report status of scan
0252 666
0252 667
0252 668 Calling sequence:
0252 669 bsbb scan
0252 670
0252 671
0252 672 Input Parameters:
0252 673
0252 674 r11 impure area address
0252 675 r10 ifab address
0252 676 r9 irab address *** please note always irab ***
0252 677 r8 rab/fab address
0252 678
0252 679 rfa <> 0 : scan for record
0252 680 r1 1'st and 2'nd word of record's rfa
0252 681 r2 3'rd word of record's rfa
0252 682 seq f.o. offset (always positive value)
0252 683 relative f.o. always 0
0252 684 index f.o. low byte = record id
0252 685
0252 686 rfa = 0 : scan for record locked by caller
0252 687 r1 is zeroed (0)
0252 688 r2 don't care
0252 689
0252 690 Output Parameters:
0252 691
0252 692 r3 is rlb found on scan or 0 if none found in scan
0252 693 r0 internal rms status code:
0252 694
0252 695 rms$_ok_alk&^xffff record was already locked by caller
0252 696
0252 697 rms$_rnl&^xffff record not locked by caller
0252 698
0252 699 Side Effects:
0252 700
0252 701 --
0252 702
0252 703 SCAN:
0252 704 ADDL3 #IRB$L_RLB_LNK,R9,R3 : get rlb header address into r3
0252 705 TSTL R1 : owner scan
0252 706 BEQL SCAN_OWNER : branch if yes
0252 707
0252 708 :
0252 709 : Scan for record match.
0252 710
0252 711 SCANLOOP:
0252 712 ASSUME RLB$L_LNK EQ 0
0252 713 MOVL (R3),R3 : get next rlb in list
0252 714 BEQL NOTFOUND : branch if at end of list
0252 715 CMPL RLB$L_RFA0(R3),R1 : compare vbn/rec#/start vbn
```

53 59 38 C1 0252 704  
51 D5 0252 705  
30 13 0252 706  
025A 707  
025A 708  
025A 709  
025A 710  
025A 711  
025A 712  
53 63 D0 025A 713  
25 13 025D 714  
51 0C A3 D1 025F 715

```
F5 12 0263 716 BNEQ SCANLOOP ; branch if no match
      0265 717
      0265 718
      0265 719 : Scans for sequential, relative, and index sequential file organization.
      0265 720 : Note: For relative file organization only need to match record number.
      0265 721
      0265 722
      0265 723 CASE TYPE=B, SRC=IFB$B_ORGCASE(R10), DISPLIST=<SCANSEQ, FOUND, SCANIDX>
      0270 724
      0270 725
      0270 726 : Scan for sequential file organization.
      0270 727
      0270 728 SCANSEQ:
      0270 729 CMPW RLBSW_RFA4(R3), R2 ; compare offset
      08 13 0274 730 BEQL FOUND ; branch if match
      E2 11 0276 731 BRB SCANLOOP ; otherwise, loop back for next
      0278 732
      0278 733
      0278 734 : Scan for indexed file organization.
      0278 735
      0278 736 SCANIDX:
      0278 737 CMPW RLBSW_ID(R3), R2 ; compare id
      DC 12 027C 738 BNEQ SCANLOOP ; branch if no match
      027E 739
      027E 740
      027E 741 : Match has been found - report status.
      027E 742
      027E 743 FOUND:
      027E 744 RMSSUC OK_ALK ; ref tag
      05 0283 745 RSB ; OK_ALK if caller already owns lock
      0284 746 ; and return
      0284 747
      0284 748 : No match found.
      0284 749
      0284 750 NOTFOUND:
      0284 751 RMSERR RNL ; set status and return
      05 0289 752 RSB
      028A 753
      028A 754
      028A 755 : Scan rlb list for owner match.
      028A 756
      028A 757 SCAN_OWNER:
      028A 758 ASSUME RLBSL_LNK EQ 0
      53 63 D0 028A 759 MOVL (R3), R3 ; get next rlb in list
      F5 13 028D 760 BEQL NOTFOUND ; branch if at end of list
      10 A3 D5 028F 761 TSTL RLBSL_OWNER(R3) ; is RLB in use?
      F6 13 0292 762 BEQL SCAN_OWNER ; branch if not
      0294 763 RMSSUC OK_ACK ; set status and return
      05 0299 764 RSB
```



```
029A 766      .SBTTL  RUSCAN
029A 767
029A 768      ++
029A 769      RUSCAN
029A 770      Look for RLB in RU lock list
029A 771
029A 772
029A 773      Calling sequence:
029A 774      bsbb      ruscan
029A 775
029A 776
029A 777      Input Parameters:
029A 778
029A 779      r11      impure area address
029A 780      r10      ifab address
029A 781      r9       irab address *** please note always irab ***
029A 782      r8       rab/fab address
029A 783      r1       1'st and 2'nd word of record's rfa
029A 784      r2       3'rd word of record's rfa
029A 785      seq f.o.      offset (always positive value)
029A 786      relative f.o. always 0
029A 787      index f.o.      low byte = record id
029A 788
029A 789      Output Parameters:
029A 790
029A 791      r3 is rlb found on scan
029A 792      r0
029A 793      RMSSUC - RLB found
029A 794      0      - RLB not found
029A 795      any error code - Returned from $ENQ service.
029A 796
029A 797      Side Effects:
029A 798
029A 799      --
029A 800
0244 31 029A 801 ERRS:  BRW      ERRENQ
029D 802
029D 803
029D 804 RUSCAN:
029D 805      PUSH  #*M<R1,R2>      ; save R1,R2
029F 806      BSBW  FLB_SCAN      ; find FLB for this file
02A2 807      BLBC  R0,T10$      ; error if none
02A5 808      CLRL  R0           ; assume failure
02A7 809      MOVAL FLB$L_RLB_LNK(R1),R2 ; get first RLB
02AB 810
02AB 811 10$:  MOVL  (R2),R2      ; next RLB into R2
02AE 812      BEQL  110$      ; error if none
02B0 813      CMPL  RLB$L_OWNER(R2),IRB$L_IDENT(R9) ; locked by this stream?
02B5 814      BNEQ  10$      ; branch if not
02B7 815      CMPL  RLB$L_RFA0(R2),(SP) ; compare VBN/REC#/start vbn
02B8 816      BNEQ  10$      ; branch if no match
02BD 817
02BD 818
02BD 819      Scans for sequential, relative, and index sequential file organization.
02BD 820      Note: For relative fiel organization only need to match record number.
02BD 821
02BD 822
```

```
02BD 823 CASE TYPE=B, SRC=IFB$B_ORGCASE(R10), DISPLIST=<30$, 50$, 40$>
02C8 824
02C8 825
02C8 826 : Scan for sequential file organization.
02C8 827
02C8 828
04 AE 06 A2 B1 02C8 829 30$: CMPW RLB$W_RFA4(R2), 4(SP) : compare offset
DC 12 02CD 830 BNEQ 10$ : branch if no match
07 11 02CF 831 BRB 50$ : match found
02D1 832
02D1 833
02D1 834 : Scan for indexed file organization
02D1 835
02D1 836
04 AE 06 A2 B1 02D1 837 40$: CMPW RLB$W_ID(R2), 4(SP) : compare id
D3 12 02D6 838 BNEQ 10$ : branch if no match
02D8 839
02D8 840 : RLB found, remove from RU RLB list, insert in IRB RLB list
02D8 841
02D8 842
02D8 843
00A8 30 02D8 844 50$: BSBW GET_RLB : get a RLB
3E BB 02D8 845 PUSHR #^MZR1, R2, R3, R4, R5 : save registers
04 A3 04 A2 18 28 02DD 846 MOVC3 #RLB$C_BLN-4, 4(R2), 4(R3) : copy saved RLB
53 04 AE D0 02E3 847 MOVL 4(SP), R3 : get saved RLB address
00E9 30 02E7 848 BSBW RESET_RLB : clear it out
3E BA 02EA 849 POPR #^M<RT, R2, R3, R4, R5> : restore registers
0B A3 21 CA 02EC 850 BICL #<RLB$M_WAIT, RLB$M_LV2>, RLB$B_FLAGS(R3) : clear WAIT, LV2
05 68 2C E1 02F0 851 BBC #RAB$V_LV2+ROP, (R8), 60$ : branch if no LV2 now
02F4 852 SSB #RLB$V_LV2, RLB$B_FLAGS(R3) : set LV2
02F9 853
02F9 854 : Find out if we have the lock in the right mode
02F9 855
02F9 856
05 68 22 E1 02F9 857 60$: BBC #RAB$V_REA+ROP, (R8), 70$ : branch if not REA lock
OE OB A3 03 E3 02FD 858 BBCS #RLB$V_PR, RLB$B_FLAGS(R3), 90$ : branch if not already PR
0302 859
05 68 33 E1 0302 860 70$: BBC #RAB$V_RLK+ROP, (R8), 80$ : branch if not RLK lock
05 OB A3 02 E3 0306 861 BBCS #RLB$V_PW, RLB$B_FLAGS(R3), 90$ : branch if not already PW
030B 862
80$: RMSSUC : set success
14 11 030E 863 BRB 110$ : lock in correct mode, get
0310 864
0310 865
0310 866 : We have to change the mode of the lock
0310 867
0310 868
05 68 31 E1 0310 869 90$: BBC #RAB$V_WAT+ROP, (R8), 100$ : branch if not wait
0314 870 SSB #RLB$V_WAIT, RLB$B_FLAGS(R3) : set wait
0319 871 100$: SSB #RLB$V_CONV, RLB$B_FLAGS(R3) : set convert
FE05 30 031E 872 BSBW DO_ENQ : go do ENQ
FDF1 30 0321 873 BSBW RRC : set codes
0324 874
06 BA 0324 875 110$: POPR #^M<R1, R2> : restore R1, R2
05 50 E9 0326 876 BLBC R0, 120$ : get out on error
0329 877 RMSSUC OK_RULK : alternate success code
032E 878
05 032E 879 120$: RSB
```

```

      032F 881      .SBTTL FLB_SCAN
      032F 882      :++
      032F 883      : FLB_SCAN
      032F 884      : Search for an FLB which matches the current IFB address
      032F 885      :--
      032F 886
      032F 887 FLB_SCAN:
51 00000000'9F D4 032F 888      CLRL R0      ; assume failure
      DE 0331 889      MOVAL @#PIOSGL_RULOCK,R1 ; get FLB list
      0338 890
      51 61 D0 0338 891 10$: MOVL (R1),R1      ; get next FLB
      OE 13 033B 892      BEQL 20$           ; get out if none
      OC A1 5A D1 033D 893      CMPL R10,FLB$L_IFB_PTR(R1) ; see if this IFB
      F5 12 0341 894      BNEQ 10$           ; branch if not
      10 A1 D5 0343 895      TSTL FLB$L_LOCK_ID(R1) ; saved file lock here?
      F0 12 0346 896      BNEQ 10$           ; skip it if so.
      0348 897      RMSSUC
      0348 898
      05 034B 899 20$: RSB

```



```
034C 901 .SBTTL PRSCAN
034C 902
034C 903 ++
034C 904 PRSCAN
034C 905 Look for RLB in RU lock list, disregarding stream, not returning lock
034C 906
034C 907
034C 908 Calling sequence:
034C 909 bsbb prscan
034C 910
034C 911
034C 912 Input Parameters:
034C 913
034C 914 r11 impure area address
034C 915 r10 ifab address
034C 916 r9 irab address *** please note always irab ***
034C 917 r8 rab/fab address
034C 918 r1 1'st and 2'nd word of record's rfa
034C 919 r2 3'rd word of record's rfa
034C 920 seq f.o. offset (always positive value)
034C 921 relative f.o. always 0
034C 922 index f.o. low byte = record id
034C 923
034C 924 Output Parameters:
034C 925
034C 926 r0
034C 927 RMSSUC - RLB found
034C 928 0 - RLB not found
034C 929
034C 930 Side Effects:
034C 931
034C 932 --
034C 933 PRSCAN: PUSH R2 ; save R2
034C 934 BSBB FLB_SCAN ; get FLB address
034C 935 BLBC R0,60$ ; get out if none
034C 936 MOVAL FLB$L_RLB_LNK(R1),R2 ; get pointer to RLBs
034C 937 CLRL R0 ; assume failure
034C 938
034C 939 10$: MOVL (R2),R2 ; next RLB into R2
034C 940 BEQL 60$ ; get out if none
034C 941 CMPL RLB$L_RFA0(R2),R1 ; compare VBN/REC#/start vbn
034C 942 BNEQ 10$ ; branch if no match
034C 943
034C 944
034C 945 : Scans for sequential, relative, and index sequential file organization.
034C 946 : Note: for relative file organization only need to match record number.
034C 947 :
034C 948
034C 949 CASE TYPE=B,SRC=IFB$B_ORGCASE(R10),DISPLIST=<30$,50$,40$>
034C 950
034C 951 :
034C 952 : Scan for sequential file organization
034C 953 :
034C 954
034C 955 30$: CMPW RLB$W_RFA4(R2),(SP) ; compare offset
034C 956 BNEQ 10$ ; branch if no match
034C 957 BRB 50$ ; match found
```

				0377	958				
				0377	959	:			
				0377	960	:	Scan for indexed file organization		
				0377	961	:			
				0377	962	:			
6E	06	A2	B1	0377	963	40\$:	CMW	RLB\$W_ID(R2),(SP)	: compare id
		DC	12	037B	964		BNEQ	10\$	: branch if no match
				037D	965				
				037D	966				
				037D	967	50\$:	RMSSUC		
				0380	968				
	04	BA		0380	969	60\$:	POPR	#^M<R2>	
		05		0382	970		RSB		

```
0383 972 .SBTTL GET_RLB AND RESET_RLB
0383 973 **
0383 974 GET_RLB - find an available rlb, if none available allocate one
0383 975 RESET_RLB - clear the RLB and indicate its free
0383 976
0383 977 Calling Sequence:
0383 978 bsbb get_rlb
0383 979 bsbb reset_rlb
0383 980
0383 981 Input Parameters:
0383 982
0383 983 get_rlb:
0383 984 r10 ifab address
0383 985 r9 irab address
0383 986 r1 1'st and 2'nd word of record's rfa
0383 987 r2 3'rd word of record's rfa
0383 988 seq f.o. offset (always positive value)
0383 989 relative f.o. always 0
0383 990 index f.o. low byte = record id
0383 991
0383 992 Output Parameters:
0383 993
0383 994 get_rlb:
0383 995 r3 points to RLB if success, else zero
0383 996
0383 997 r0 internal RMS status code:
0383 998 DME - couldn't allocate an RLB
0383 999 SUC - r3 points to RLB
0383 1000
0383 1001 Side Effects:
0383 1002
0383 1003 If success, RLB owner and RFA fields initialized.
0383 1004 --
0383 1005
0383 1006
0383 1007 Record is not in local list of locked records, so scan the rlb list for
0383 1008 an available rlb.
0383 1009
0383 1010
0383 1011 GET_RLB:
0383 1012 ADDL3 #IRB$RLB_LNK,R9,R3 ; find an rlb
0383 1013 ASSUME RLB$RLB_LNK EQ 0 ; get rlb header address
0383 1014 10$: MOVL (R3),R3 ; get next rlb in list
0383 1015 BEQL 20$ ; if eql end of list, go allocate one
0383 1016 TSTL RLB$RLB_OWNER(R3) ; is rlb available
0383 1017 BNEQ 10$ ; loop back for next if not
0383 1018 BRB 30$ ; success, r3 points to rlb
0383 1019
0383 1020
0383 1021 No available rlb so we must allocate a new one.
0383 1022
0383 1023 20$: PUSHR #^M<R1,R2,R4> ; save registers
0383 1024 MOVL R10,R1 ; set addr in page = ifab
0383 1025 MOVZBL #RLB$C_BLN/4,R2 ; set # of long words
0383 1026 BSBW RMSGETBLK ; get rlb block and fill in length
0383 1027 MOVL R1,R3 ; copy address if any
0383 1028 POPR #^M<R1,R2,R4> ; restore registers
```

53	59	38	C1	0383	1012	ADDL3	#IRB\$RLB_LNK,R9,R3	; find an rlb
				0383	1013	ASSUME	RLB\$RLB_LNK EQ 0	; get rlb header address
53	63	D0		0383	1014	10\$: MOVL	(R3),R3	; get next rlb in list
	07	13		038A	1015	BEQL	20\$	; if eql end of list, go allocate one
10	A3	D5		038C	1016	TSTL	RLB\$RLB_OWNER(R3)	; is rlb available
	F6	12		038F	1017	BNEQ	10\$	; loop back for next if not
	29	11		0391	1018	BRB	30\$	; success, r3 points to rlb
				0393	1019			
				0393	1020			
				0393	1021			
				0393	1022			
	16	BB		0393	1023	20\$: PUSHR	#^M<R1,R2,R4>	; save registers
51	5A	D0		0395	1024	MOVL	R10,R1	; set addr in page = ifab
52	07	9A		0398	1025	MOVZBL	#RLB\$C_BLN/4,R2	; set # of long words
	FC62'	30		039B	1026	BSBW	RMSGETBLK	; get rlb block and fill in length
53	51	D0		039E	1027	MOVL	R1,R3	; copy address if any
	16	BA		03A1	1028	POPR	#^M<R1,R2,R4>	; restore registers



```

      27 50   E9 03A3 1029      BLBC      R0,ERRDME      ; if we failed then exit
      070E 8F   B0 03A6 1030      ASSUME     RLB$B_BLN EQ RLB$B_BID+1
      08 A3     B0 03A6 1031      MOVW      #RLB$C_BID+<RLB$C_BLN@6>,-
      50 53     D0 03AA 1032      RLB$B_BID(R3)      ; set block id code
      53 38     C1 03AC 1033      MOVL      R3,R0      ; save new rlb address
      59 38     C1 03AF 1034      ADDL3     #IRB$L_RLB_LNK,R9,R3 ; get rlb header address
      60 63     D0 03B3 1035      ASSUME     RLB$L_LNK EQ 0
      63 50     D0 03B3 1036      MOVL      (R3),(R0)    ; set ptr to next in new rlb
      53 50     D0 03B6 1037      MOVL      R0,(R3)    ; put new rlb at front of list
      60 63     D0 03B9 1038      MOVL      R0,R3      ; restore new rlb address
      63 50     D0 03BC 1039      30$:          ; initialize RLB
      53 50     D0 03BC 1040      SETOWNRFA:      ; can be called here by QUERY_LCK
      34 A9     D0 03BC 1041      MOVL      IRB$L_IDENT(R9),-
      10 A3     D0 03BF 1042      RLB$L_OWNER(R3)      ; set owner isi
      0C A3     D0 03C1 1043      MOVL      R1,RLB$L_RFA0(R3) ; set records rfa in rlb
      06 A3     B0 03C5 1044      MOVW      R2,RLB$W_RFA4(R3)
      05 05     05 03C9 1045      RMSSUC
      05 05     05 03CC 1046      RSB          ; indicate success
      05 05     05 03CD 1047      RSB          ; and return
      05 05     05 03CD 1048      ;
      05 05     05 03CD 1049      ; error allocating rlb
      05 05     05 03CD 1050      ;
      05 05     05 03CD 1051      ERRDME: RMSERR DME      ; no dynamic memory
      05 05     05 03D2 1052      RSB          ; return to caller
      05 05     05 03D3 1053      ;
      05 05     05 03D3 1054      ;
      05 05     05 03D3 1055      ;++
      05 05     05 03D3 1056      ; RESET_RLB
      05 05     05 03D3 1057      ;
      05 05     05 03D3 1058      ; Indicate the RLB is free, and clean it up. Called from UNLOCK and errors
      05 05     05 03D3 1059      ; on LOCK.
      05 05     05 03D3 1060      ;
      05 05     05 03D3 1061      ; r0 must be preserved by this routine
      05 05     05 03D3 1062      ; r3 points to the RLB
      05 05     05 03D3 1063      ;
      05 05     05 03D3 1064      ; Note that RLB$B_TMO is not cleared, since it is only meaningful when
      05 05     05 03D3 1065      ; RLB$V_TMO is set in RLB$B_FLAGS, which is cleared here.
      05 05     05 03D3 1066      ;--
      05 05     05 03D3 1067      ;
      05 05     05 03D3 1068      ; ASSUME <RLB$L_RFA0+4> EQ RLB$L_OWNER
      05 05     05 03D3 1069      RESET_RLB:
      04 A3     D4 03D3 1070      CLRL      RLB$L_MISC(R3)      ; Clears RLB$W_FLAGS2.
      0C A3     7C 03D6 1071      CLRQ      RLB$L_RFA0(R3)
      08 A3     94 03D9 1072      CLRB      RLB$B_FLAGS(R3)
      05 05     05 03DC 1073      RSB          ; return to caller
```

```
03DD 1075      .SBTTL  RMSUNLOCK AND RMSUNLOCKALL
03DD 1076
03DD 1077      :++
03DD 1078      :RMSUNLOCK
03DD 1079      :RMSUNLOCKALL
03DD 1080
03DD 1081      :Deletes entries in the record lock list
03DD 1082
03DD 1083      :RMSUNLOCK_HARD
03DD 1084
03DD 1085      :Deletes an entry in the record lock list, but maps a REA lock held by
03DD 1086      :the caller to RNL so a writer holding a REA lock does not attempt an
03DD 1087      :update or delete.
03DD 1088
03DD 1089      :Calling sequence:
03DD 1090      :           bsbw      rm$unlock
03DD 1091      :           bsbw      rm$unlockall
03DD 1092      :           bsbw      rm$unlock_hard
03DD 1093
03DD 1094      :Input Parameters:
03DD 1095
03DD 1096      :           r11      impure area address
03DD 1097      :           r10      ifab (shared ifab) address
03DD 1098      :           r9       irab address *** please note always irab ***
03DD 1099      :           r8       rab/fab address
03DD 1100
03DD 1101      :           rfa <> 0 :      unlock record
03DD 1102      :           r1       1'st and 2'nd word of record's rfa
03DD 1103      :           r2       3'rd word of record's rfa
03DD 1104      :           seq f.o.   offset (always positive value)
03DD 1105      :           relative f.o. always 0
03DD 1106      :           index f.o.  low byte = record id
03DD 1107
03DD 1108      :           rm$unlockall entry
03DD 1109      :           r1,r2 don't care
03DD 1110
03DD 1111      :Output Parameters:
03DD 1112
03DD 1113      :           r3 is destroyed
03DD 1114
03DD 1115      :           r0       internal rms status code
03DD 1116      :           rms$_suc&^xffff record(s) unlocked
03DD 1117      :           rms$_rnl&^xffff record was not locked
03DD 1118      :                           or no record was locked (unlock all call)
03DD 1119
03DD 1120      :           rm$unlockall:
03DD 1121      :           the irb$_unlock_rp irab bookkeeping bit is cleared
03DD 1122      :           r1 is zeroed
03DD 1123
03DD 1124      :Side Effects:
03DD 1125
03DD 1126      :--
03DD 1127
03DD 1128
03DD 1129      :RMSUNLOCK_HARD::
03DD 1130      :           BSBW      SCAN      ; find record
50      FE72      30      03DD 1130      CMPW      #<RMS$_OK_ALK&^XFFFF>,R0; caller locked record?
      8039 8F      B1      03E0 1131
```

```

      OC 12 03E5 1132      BNEQ 10$      ; if neq no, return RNL error
      03 E1 03E7 1133      BBC          ; did caller lock record REA?
18 OB A3 10 03E9 1134      BSBB      RLB$B_FLAGS(R3),DEQUE ; no, continue usual path
      16 03EC 1135      RMSERR DEQUE-    ; yes, go unlock the sucker and...
      05 03EE 1136      RSB          ; return RNL so no update is attempted
      03F3 1137 10$:      ; return to caller
      03F4 1138
      03F4 1139 DEQUE_QUERY:      ; called here from QUERY
25 6A 33 E1 03F4 1140      BBC          ; do a deq if record locking
      36 11 03F8 1141      BRB          DEQ_RS      ; go release RLB
      03FA 1142
      03FA 1143 RMSUNLOCK::      ;
      03FA 1144 UNLOCK:          ;
      50 FE55 30 03FA 1145      BSBW SCAN      ; ref tag
      8039 8F B1 03FD 1146      CMPW #<RMS$ OK_ALKB^XFFFF>,R0 ; scan for record
      32 12 0402 1147      BNEQ NOTLOCK ; did we find a locked record for stream
      0404 1148      ; branch if no
      0404 1149 :++
      0404 1150 : Perform the $DEQ_S
      0404 1151 :
      0404 1152 :
      0404 1153 :--
      0404 1154
      04 06 E0 0404 1155 DEQUE: BBS #RLB$V_FAKE,-      ; if 'fake' RLB then maybe RUSAVE
      A3 33 E0 0406 1156      BBS      RLB$B_FLAGS(R3),10$ ;
      23 6A 0409 1157      BBS      #IFB$V_NORECLK,-      ; dont do a deq if no record locking
      040B 1158      (R10),DEQ_RS
      040D 1159
      0A 00A2 02 E1 040D 1160 10$: BBC #IFB$V_RUP,-      ;
      CA 040F 1161      IFB$B_JNLFLG2(R10),DEQ ; branch if not in RU
      05 E1 0413 1162      BBC      #RLB$V_LV2,-      ;
      24 OB A3 0415 1163      RLB$B_FLAGS(R3),RUSAVE ; save lock if not Level 2
      03 E1 0418 1164      BBC      #RLB$V_PR,-      ; if level 2 save all but
      1F OB A3 041A 1165      RLB$B_FLAGS(R3),RUSAVE ; PR locks.
      OE OB A3 06 E0 041D 1166 DEQ: BBS #RLB$V_FAKE,RLB$B_FLAGS(R3),DEQ_RS ; branch if fake RLB
      0422 1167      $DEQ_S LKID=RLB$L_LOCK_ID(R3) ; lock id of lock to unlock
      0430 1168      ; ignore errors...
      0430 1169
      A1 10 0430 1170 DEQ_RS: BSBB RESET_RLB      ; free the rlb
      0432 1171      RMSSUC      ; say success
      05 0435 1172      RSB          ; and return
      0436 1173
      0436 1174 NOTLOCK:      ;
      0436 1175      RMSERR RNL      ; say record not locked
      05 043B 1176      RSB          ; and exit
      043C 1177
      043C 1178 :
      043C 1179 : Save locks given up in Recovery Units
      043C 1180 :
      043C 1181 :
      043C 1182 ASSUME <RLB$C_BLN+1> GT FLB$C_BLN
      043C 1183
      3E BB 043C 1184 RUSAVE: PUSHF #^M<R1,R2,R3,R4,R5> ; save registers
      FEEE 30 043E 1185      BSBW FLB_SCAN      ; see if there is already an FLB
      1B 50 E8 0441 1186      BLBS R0,T0$      ; branch if so
      4E 10 0444 1187      BSBB ALOCPBLK      ; get an FLB
      3F 50 E9 0446 1188      BLBC R0,50$      ; get out on error
```

```
61 00000000'9F D0 0449 1189 MOVL @#PIO$GL_RULOCK,(R1) ; set successor to first FLB
00000000'9F 51 D0 0450 1190 MOVL R1,@#PIO$GL_RULOCK ; set new FLB as first FLB
08 A1 17 90 0457 1191 MOVVB #FLB$C_BID,FLB$B_BID(R1) ; set block id in FLB
0C A1 5A D0 045B 1192 MOVL R10,FLB$L_IFB_PTR(R1) ; set IFAB address in FLB
045F 1193
52 51 D0 045F 1194 10$: MOVL R1,R2 ; save FLB address
51 04 A2 DE 0462 1195 MOVAL FLB$L_RLB_LNK(R2),R1 ; get RLB pointer
0466 1196
51 61 D0 0466 1197 20$: MOVL (R1),R1 ; get RLB
07 13 0469 1198 BEQL 30$ ; branch if none
10 A1 D5 046B 1199 TSTL RLB$L_OWNER(R1) ; is RLB available
F6 12 046E 1200 BNEQ 20$ ; branch if not
0D 11 0470 1201 BRB 40$ ; go use it otherwise
0472 1202
20 10 0472 1203 30$: BSBB ALOCPBLK ; get an RLB
11 50 E9 0474 1204 BLBC R0,50$ ; get out on error
61 04 A2 D0 0477 1205 MOVL FLB$L_RLB_LNK(R2),(R1) ; set successor to first RLB
04 A2 51 D0 047B 1206 MOVL R1,FLB$L_RLB_LNK(R2) ; set new RLB as first RLB
047F 1207
04 A1 04 A3 18 28 047F 1208 40$: MOVC3 #RLB$C_BLN-4,4(R3),4(R1); copy old RLB
0485 1209 RMSSUC
0488 1210
3E BA 0488 1211 50$: POPR #^M<R1,R2,R3,R4,R5>
FF46 30 048A 1212 BSBW RESET_RLB ; remove lock from IFB RLB list
03 50 E8 048D 1213 BLBS R0,60$ ; return on success
FF3A 31 0490 1214 BRW ERRDME ; only error possible is DME
05 0493 1215 60$: RSB
0494 1216
0494 1217 ; ALOCPBLK - Allocate a block for the RULOCK list
0494 1218 ;
0494 1219 ;
0494 1220 ;
0494 1221 ALOCPBLK:
083C 8F BB 0494 1222 PUSHR #^M<R2,R3,R4,R5,R11> ; save registers
5B 00000000'9F DE 0498 1223 MOVAL @#PIO$GW_PIOIMPA,R11 ; PIO free list header
51 5B D0 049F 1224 MOVL R11,R1
52 07 9A 04A2 1225 MOVZBL #RLB$C_BLN/4,R2 ; set # of long words
FB58' 30 04A5 1226 BSBW RM$GETBLK ; get block
083C 8F BA 04A8 1227 POPR #^M<R2,R3,R4,R5,R11> ; restore registers
05 04AC 1228 RSB
04AD 1229
04AD 1230 ; DEAPBLK - Deallocate a RULOCK block
04AD 1231 ;
04AD 1232 ;
04AD 1233 ;
04AD 1234 DEAPBLK:
083C 8F BB 04AD 1235 PUSHR #^M<R2,R3,R4,R5,R11>
5B 00000000'9F DE 04B1 1236 MOVAL @#PIO$GW_PIOIMPA,R11 ; PIO free list header
53 5B D0 04B8 1237 MOVL R11,R3
54 51 D0 04BB 1238 MOVL R1,R4
FB3F' 30 04BE 1239 BSBW RM$RETBLK ; return space
083C 8F BA 04C1 1240 POPR #^M<R2,R3,R4,R5,R11>
05 04C5 1241 RSB
04C6 1242
04C6 1243 ;
04C6 1244 ; Unlock all records for the caller.
04C6 1245 ;
```



		04C6	1246						
		04C6	1247	RMSUNLOCKALL::					
		04C6	1248	CSB	#IRBSV_UNLOCK_RP,(R9)				
		04CA	1249						
51	D4	04CA	1250	CLRL	R1				
FF2B	30	04CC	1251	BSBW	UNLOCK				
OC 50	E9	04CF	1252	BLBC	RO,NTLK				
51	D4	04D2	1253	10\$: CLRL	R1				
FF23	30	04D4	1254	BSBW	UNLOCK				
FB 50	E8	04D7	1255	BLBS	RO,10\$				
		04DA	1256	RMSSUC					
	05	04DD	1257	RSB					
		04DE	1258						
FF55	31	04DE	1259	NTLK: BRW	NOTLOCK				
		04E1	1260						
		04E1	1261	ERRENO: RMSPPBUG FTL\$_ENQDEQFAIL					

rp will be unlocked so note that  
it was done  
flag owner scan  
unlock first record  
if we failed then exit  
re-flag owner scan, since DEQ blew R1  
unlock next record  
loop back if success  
exit with success

the deq failed

```

04E8 1263      .SBTTL RMSSAVE_FL
04E8 1264      :++
04E8 1265      : RMSSAVE_FL - Save the file lock in the RULOCK List
04E8 1266      :
04E8 1267      : Calling sequence:
04E8 1268      :         BSBW      RMSSAVE_FL
04E8 1269      :
04E8 1270      : Input Parameters:
04E8 1271      :         R4      -      SFSB address
04E8 1272      :         R9      -      IFAB address
04E8 1273      :
04E8 1274      : Output Parameters:
04E8 1275      :         R1      -      Destroyed
04E8 1276      :
04E8 1277      : Side Effects:
04E8 1278      :         None.
04E8 1279      :
04E8 1280      :--
04E8 1281
04E8 1282      RMSSAVE_FL::
04E8 1283      PUSHL      R10
04E8 1284      MOVL      R9,R10
04E8 1285      BSBW      FLB_SCAN
04E8 1286      BLBS      R0,T0$
04E8 1287      BSBB      ALOCPLK
04E8 1288      BLBC      R0,20$
04E8 1289      MOVL      @#PIO$GL RULOCK,(R1)
04E8 1290      MOVL      R1,@#PIO$GL RULOCK
04E8 1291      MOVB      #FLB$C BID,FLB$B BID(R1)
04E8 1292      MOVL      R9,FLB$L_IFB_PTR(R1)
04E8 1293
04E8 1294      10$:      MOVL      SFSB$L_LOCK_ID(R4),FLB$L_LOCK_ID(R1) ; save lock
04E8 1295
04E8 1296      20$:      POPL      R10
04E8 1297      RSB

```

5A DD 04E8 1283  
 5A 59 D0 04EA 1284  
 FE3F 30 04ED 1285  
 1B 50 E8 04F0 1286  
 9F 10 04F3 1287  
 1B 50 E9 04F5 1288  
 61 00000000'9F D0 04F8 1289  
 00000000'9F 51 D0 04FF 1290  
 08 A1 17 90 0506 1291  
 0C A1 59 D0 050A 1292  
 10 A1 30 A4 D0 050E 1293  
 5A 8ED0 0513 1294  
 05 0516 1297

: save R10  
 : move IFB address for FLB\_SCAN  
 : see if there is an FLB  
 : branch if one  
 : get an FLB  
 : get out on error  
 : set successor to first FLB  
 : set new FLB as first FLB  
 : set block id in FLB  
 : set IFAB address in FLB

```
0517 1299 .SBTTL RMSRU_UNLOCK
0517 1300
0517 1301 :++
0517 1302 RMSRU_UNLOCK - Unlocks all locks held for recovery unit
0517 1303 :
0517 1304 Calling sequence:
0517 1305 BSBW RMSRU_UNLOCK
0517 1306 :
0517 1307 Input Parameters:
0517 1308 None.
0517 1309 :
0517 1310 Output Parameters:
0517 1311 R0-R5 Destroyed
0517 1312 :
0517 1313 Side Effects:
0517 1314 None.
0517 1315 :
0517 1316 :--
0517 1317
0517 1318 RMSRU_UNLOCK::
58 00000000'9F DD 0517 1319 PUSHL R11 ; save impure area address
53 00000000'9F DE 0519 1320 MOVAL @#PIO$GW_PIOIMPA,R11 ; point to process I/O set
53 00000000'9F DO 0520 1321 MOVL @#PIO$GL_RULOCK,R3 ; get first FLB
53 00000000'9F 0527 1322
54 04 A3 13 0527 1323 10$: BEQL 60$ ; branch if none
54 04 A3 DO 0529 1324 MOVL FLB$L_RLB_LNK(R3),R4 ; get RLB address
54 04 A3 052D 1325
54 04 A3 13 052D 1326 20$: BEQL 40$ ; branch if at end of list
54 04 A3 052F 1327 $DEQ S LKID=RLB$L_LOCK_ID(R4) ; deque record lock
51 54 DD 053D 1328 PUSHC (R4) ; save next RLB address
51 54 DO 053F 1329 MOVL R4,R1 ; return RLB
54 FF68 30 0542 1330 BSBW DEAPBLK
54 8E DO 0545 1331 MOVL (SP)+,R4 ; get next RLB
54 E3 11 0548 1332 BRB 20$ ; go process next RLB
54 E3 054A 1333
54 10 A3 D5 054A 1334 40$: TSTL FLB$L_LOCK_ID(R3) ; was a file lock saved?
54 10 A3 13 054D 1335 BEQL 50$ ; branch if not
54 10 A3 054F 1336 $DEQ S LKID=FLB$L_LOCK_ID(R3) ; deque file lock
51 63 DD 055D 1337 50$: PUSHC (R3) ; save next FLB address
51 63 DO 055F 1338 MOVL R3,R1 ; return FLB
51 FF48 30 0562 1339 BSBW DEAPBLK
53 8E DO 0565 1340 MOVL (SP)+,R3 ; get next FLB address
53 BD 11 0568 1341 BRB 10$ ; go process it
53 BD 056A 1342
53 00000000'9F D4 056A 1343 60$: CLRL @#PIO$GL_RULOCK ; clear list header
53 5B 8E DO 0570 1344 MOVL (SP)+,R11 ; restore R11
53 05 0573 1345 RSB
53 0574 1346
53 0574 1347 .END
```

\$\$PSECT\_EP  
\$\$ARGS  
\$\$RMSTEST  
\$\$RMS\_PBUGCHK  
\$\$RMS\_TBUGCHK  
\$\$RMS\_UMODE  
\$\$T1  
ALOCBLK  
DEAPBLK  
DEQ  
DEQUE  
DEQUE\_QUERY  
DEQ\_RS  
DO\_ENQ  
ENQS\_ACMODE  
ENQS\_ASTADR  
ENQS\_ASTPRM  
ENQS\_BLKAST  
ENQS\_EFN  
ENQS\_FLAGS  
ENQS\_LKMODE  
ENQS\_LKSB  
ENQS\_NARGS  
ENQS\_PARID  
ENQS\_PROT  
ENQS\_RESNAM  
ERRDME  
ERRENO  
ERRS  
FLBSB\_BID  
FLBSC\_BID  
FLBSC\_BLN  
FLBSL\_IFB\_PTR  
FLBSL\_LOCK\_ID  
FLBSL\_RLB\_LNK  
FLB\_SCAN  
FOUND  
FTLS\_ENQDEQFAIL  
FTLS\_NOCURBDB  
GET\_RLB  
IFBSB\_JNLFLG  
IFBSB\_JNLFLG2  
IFBSB\_ORGCASE  
IFBSB\_RECVFLGS  
IFBSL\_SFSB\_PTR  
IFBSM\_ONLY\_RU  
IFBSM\_RU  
IFBSV\_NORECLK  
IFBSV\_RUP  
IFBSV\_RU\_RECVR  
IFBSV\_RU\_RLK  
IRBSB\_EFN  
IRBSL\_CURBDB  
IRBSL\_IDENT  
IRBSL\_RLB\_LNK  
IRBSV\_NO\_WAIT  
IRBSV\_UNLOCK\_RP

= 00000000  
= 0000000B  
= 0000001A  
= 00000010  
= 00000008  
= 00000004  
= 00000001  
00000494 P 01  
000004AD R 01  
0000041D R 01  
00000404 R 01  
000003F4 R 01  
00000430 R 01  
00000126 R 01  
= 00000028  
= 0000001C  
= 00000020  
= 00000024  
= 00000004  
= 00000010  
= 00000008  
= 0000000C  
= 0000000B  
= 00000018  
= 0000002C  
= 00000014  
000003CD R 01  
000004E1 R 01  
0000029A R 01  
= 00000008  
= 00000017  
= 00000014  
= 0000000C  
= 00000010  
= 00000004  
0000032F R 01  
0000027E R 01  
= FFFFFFF2  
= FFFFFFF1  
00000383 R 01  
= 000000A0  
= 000000A2  
= 00000023  
= 000000A1  
= 00000078  
= 00000001  
= 00000002  
= 00000033  
= 00000002  
= 00000000  
= 00000003  
= 0000000B  
= 00000020  
= 00000034  
= 00000038  
= 00000038  
= 0000002D

LCK\$K\_CRMODE  
LCK\$K\_EXMODE  
LCK\$K\_PRMODE  
LCK\$K\_PWMODE  
LCK\$M\_CONVERT  
LCK\$M\_NOQUEUE  
LCK\$M\_SYNCSTS  
LCK\$M\_SYSTEM  
LCK\$V\_PROTECT  
LCK\$V\_RECOVER  
NOBDB  
NOTFOUND  
NOTLOCK  
NTLK  
PIO\$A\_TRACE  
PIO\$G\_RULOCK  
PIO\$GW\_PIOIMPA  
PRSCAN  
RAB\$B\_TMO  
RAB\$B\_ROP  
RAB\$V\_LV2  
RAB\$V\_REA  
RAB\$V\_RLK  
RAB\$V\_RRL  
RAB\$V\_TMO  
RAB\$V\_WAT  
RESET\_RLB  
RLB\$B\_BID  
RLB\$B\_BLN  
RLB\$B\_FLAGS  
RLB\$B\_TMO  
RLB\$C\_BID  
RLB\$C\_BLN  
RLB\$C\_LKSB  
RLB\$C\_LNK  
RLB\$C\_LOCK\_ID  
RLB\$C\_MISC  
RLB\$C\_OWNER  
RLB\$C\_RFA0  
RLB\$M\_CR  
RLB\$M\_LV2  
RLB\$M\_PR  
RLB\$M\_WAIT  
RLB\$V\_CONV  
RLB\$V\_CR  
RLB\$V\_FAKE  
RLB\$V\_LV2  
RLB\$V\_PR  
RLB\$V\_PW  
RLB\$V\_TIMER\_INPROG  
RLB\$V\_TMO  
RLB\$V\_WAIT  
RLB\$W\_FLAGS2  
RLB\$W\_ID  
RLB\$W\_RFA4  
RLB\$W\_STATUS  
RMSBUG

= 00000001  
= 00000005  
= 00000003  
= 00000004  
= 00000002  
= 00000004  
= 00000008  
= 00000010  
= 00000008  
= 00000007  
0000024B R 01  
00000284 R 01  
00000436 R 01  
000004DE R 01  
\*\*\*\*\* X 01  
\*\*\*\*\* X 01  
\*\*\*\*\* X 01  
0000034C R 01  
= 0000001F  
= 00000004  
= 0000000C  
= 00000002  
= 00000013  
= 00000003  
= 00000019  
= 00000011  
000003D3 R 01  
= 00000008  
= 00000009  
= 0000000B  
= 0000000A  
= 0000000E  
= 0000001C  
= 00000014  
= 00000000  
= 00000018  
= 00000004  
= 00000010  
= 0000000C  
= 00000002  
= 00000020  
= 00000008  
= 00000001  
= 00000004  
= 00000001  
= 00000006  
= 00000005  
= 00000003  
= 00000002  
= 00000000  
= 00000007  
= 00000000  
= 00000004  
= 00000006  
= 00000006  
= 00000014  
\*\*\*\*\* X 01



RMORECLCK  
Symbol table

RECORD LOCK LIST (RLB) PROCESSING J 12

16-SEP-1984 00:32:06 VAX/VMS Macro V04-00  
5-SEP-1984 16:22:15 [RMS.SRC]RMORECLCK.MAR;1

Page 32  
(13)

RMSGETBLK	*****	X	01
RMSLOCK	00000000	RG	01
RMSMAPERR	*****	X	01
RMSQUERY_HARD	00000070	RG	01
RMSQUERY_LCK	000000A1	RG	01
RMSQUERY_PROC	0000008B	RG	01
RMSRELEASE	*****	X	01
RMSRETBK	*****	X	01
RMSRU_UNLOCK	00000517	RG	01
RMSSAVE_FL	000004E8	RG	01
RMSSETFN	*****	X	01
RMSSET_LOCK_TMO	*****	X	01
RMSSTACL	*****	X	01
RMSSTALLAST	*****	X	01
RMSUNLOCK	000003FA	RG	01
RMSUNLOCKALL	000004C6	RG	01
RMSUNLOCK_HARD	000003DD	RG	01
RMS\$_DME	= 000184D4		
RMS\$_ENQ	= 0001C134		
RMS\$_OK_ALK	= 00018039		
RMS\$_OK_RLK	= 00018021		
RMS\$_OK_RRL	= 00018029		
RMS\$_OK_RULK	= 00018071		
RMS\$_OK_WAT	= 00018061		
RMS\$_RLK	= 000182AA		
RMS\$_RNL	= 000181A0		
RMS\$_TMO	= 000181B0		
RMS\$_TMR	= 0001C16C		
ROP	= 00000020		
RRL	00000115	R	01
RUSAVE	0000043C	R	01
RUSCAN	0000029D	R	01
SCAN	00000252	R	01
SCANIDX	00000278	R	01
SCANLOOP	0000025A	R	01
SCANSEQ	00000270	R	01
SCAN_OWNER	0000028A	R	01
SETOWNRFA	000003BC	R	01
SFSBSL_LOCK_ID	= 00000030		
SS\$_ABORT	= 0000002C		
SS\$_SYNCH	= 00000689		
SYSSCANTIM	*****	GX	01
SYSSDEQ	*****	GX	01
SYSSENQ	*****	X	01
SYSSSETFN	*****	GX	01
TPTSL_LOCK	*****	X	01
TPTSL_QUERY_LCK	*****	X	01
TPTSL_REC_WAT	*****	X	01
UNLOCK	000003FA	R	01

RMO  
V04

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 ( 0.)	00 ( 0.)	NOPIC USR
RMSRMS0	00000574 ( 1396.)	01 ( 1.)	PIC USR
\$ABSS	00000000 ( 0.)	02 ( 2.)	NOPIC USR

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.09	00:00:01.35
Command processing	107	00:00:00.69	00:00:05.06
Pass 1	415	00:00:15.67	00:00:54.17
Symbol table sort	0	00:00:01.99	00:00:04.18
Pass 2	231	00:00:04.07	00:00:14.01
Symbol table output	20	00:00:00.19	00:00:00.69
Psect synopsis output	1	00:00:00.06	00:00:00.36
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	805	00:00:22.77	00:01:19.83

The working set limit was 1650 pages.  
86780 bytes (170 pages) of virtual memory were used to buffer the intermediate code.  
There were 70 pages of symbol table space allocated to hold 1337 non-local and 91 local symbols.  
1347 source lines were read in Pass 1, producing 18 object records in Pass 2.  
36 pages of virtual memory were used to define 35 macros.

! Macro library statistics !

Macro library name	Macros defined
-\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	17
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	1
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	13
TOTALS (all libraries)	31

1493 GETS were required to define 31 macros.  
There were no errors, warnings or information messages.  
MACRO/LIS=LIS\$:RMORECLCK/OBJ=OBJ\$:RMORECLCK MSRC\$:RMORECLCK/UPDATE=(ENH\$:RMORECLCK)+EXECMLS/LIB+LIB\$:RMS/LIB



0319 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

